1 Introduction

Diabetes mellitus, commonly referred to as diabetes, is a condition characterised by elevated blood glucose levels. The body cannot produce enough insulin or is not effectively using the insulin produced. Over time, high glucose levels can damage blood vessels and nerves, resulting in long-term health complications including heart, kidney, eye and foot damage (Diabetes Australia n.d.). Hence, early detection and diagnosis are crucial, and this report proposes a method to predict the condition using a **Multi Layer Perceptron** (MLP). MLP is a type of Neural Network which consists of multiple layers of neurons with non-linear activation functions, allowing the network to learn complex patterns in data. In this work, analysis and comparison are made between different hyperparameter settings involving learning rate, activation functions and batch size, via k-fold cross-validation, and the best settings are selected for Final training and Testing on the dataset. The metrics considered are **Accuracy, Receiver Operating characteristic and Area under the curve (ROC-AUC)**. The proposed model effectively captures the underlying patterns in the medical data, achieving ROC-AUC of **0.817**. meaning it ranks positive cases correctly about 81% of the time, which is a strong indicator of good discriminative power. Furthermore, this model has achieved the training accuracy of **82.78%** and the test accuracy of **70.78%**.

2 Method

For predicting Diabetes, the dataset used is obtained from *the National Institute of Diabetes and Digestive and Kidney Diseases*. It consists of 8 Features including BloodPressure, SkinThickness, Insulin, BMI, Age, etc. For **data preprocessing**, rows in the dataset that contain outlier values based on the Interquartile Range (IQR) method are detected, where any data point lying outside 1.5 times the IQR from the 1^{st} or 3^{rd} quartile is considered an outlier. Then, the outliers are removed, reducing our sample size from 614 to 511 rows. **Standard scaling** is performed to remove the mean and scale to unit variance.

$$\boxed{8} \rightarrow \boxed{16} \rightarrow \boxed{10} \rightarrow \boxed{5} \rightarrow \boxed{2}$$

(Input → Hidden1 → Hidden2 → Hidden3 → Output)

The model supports three activation functions: **ReLU**, **GELU**, and **Leaky ReLU**. Each activation introduces non-linearity and enables the network to learn complex relationships between the input features and the diabetes outcome.

Linear Layers: Each fully connected layer applies a learned weight matrix and bias vector to project the inputs into a higher-level representation. This allows the model to capture interactions between multiple medical attributes that are not linearly separable. Leaky ReLU and GELU can retain information in these regions, improving model sensitivity.

Dropout: Dropout with $p = 0.2$ randomly deactivates neurons during training, reducing co-adaptation of neurons and improving generalization. This is particularly beneficial for smaller datasets.

Output Layer and Loss: The final layer has two outputs (logits) corresponding to the classes "non-diabetic" and "diabetic." The `CrossEntropyLoss` function is used for training, which combines the softmax operation and the negative log-likelihood loss. During evaluation, the class with the highest logit value is selected using `argmax`.

The model is implemented using PyTorch.

```
import torch.nn as nn
class Model_a2(nn.Module):
    def __init__(self, activation_fn='relu', input_features=8, dropout_prob=0.2):
```

```
self.fc1 = nn.Linear(input_features, 16)
self.fc2  = nn.Linear(16, 10)
self.fc3  = nn.Linear(10, 5)
self.fc4  = nn.Linear(5, 2)
self.act_fn = #RELU/GELU/LeakyRELU...
self.dropout = nn.Dropout(dropout_prob)...
```

Forward Pass Algorithm:

```
def forward(self, x):
    x=self.dropout(self.act_fn(self.fc1(x)))
    x=self.dropout(self.act_fn(self.fc2(x)))
    x=self.dropout(self.act_fn(self.fc3(x)))
    x=self.fc4(x)
    return x
```

Training and Backpropagation:

```
def train_one_epoch(self,model,train_loader):
 for X, y in train_loader:
     optimizer.zero_grad()
     outputs = model(X)
     loss = criterion(outputs, y)
     loss.backward()
     optimizer.step() ...
```

Limitations:

- The model is sensitive to unscaled or noisy inputs; normalization is essential.
- Interpretability is limited compared to linear or tree-based models.
- Imbalanced data can bias accuracy; therefore, metrics such as ROC-AUC or F1-score provide more reliable evaluation.

Proposed Improvements

To enhance the baseline model, several modifications can be made, including: **Batch Normalization** after each linear layer to stabilise activation distributions and accelerate training, **Class-weighted CrossEntropyLoss** to address class imbalance, **Early Stopping** based on validation AUC or loss to prevent overfitting, and **Threshold Optimization** to balance sensitivity and specificity depending on clinical priorities.

3 Experimental Analysis

For selecting the best set of hyperparameters, the following three configurations were run:

Configuration 1

```
batch_size: 16
device: cuda
optimizer: SGD
num_epochs: 10
learning_rate: 0.001
activation_fn: relu
```

Configuration 2

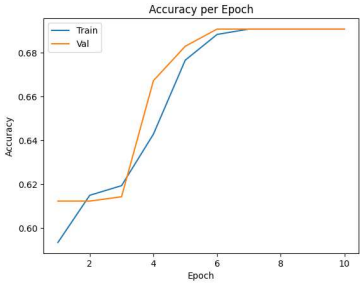
```
batch_size: 16
device: cuda
optimizer: Adam
num_epochs: 20
learning_rate: 0.01
activation_fn: leaky_relu
```

Configuration 3

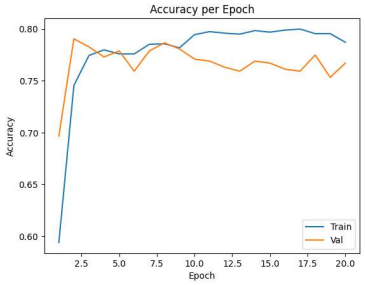
```
batch_size: 32
device: cuda
optimizer: AdamW
num_epochs: 20
learning_rate: 0.01
activation_fn: gelu
```

In the experiment, **Configuration 3** was found to be the best with highest Accuracy and the least loss, after 5 folds of Cross-validation. With these hyperparameters, final training was performed for 20 epochs on the full dataset with shape: (511, 9), which achieved training accuracy of 0.8278 at the final epoch, followed by Final testing on the held-out Test dataset, achieving **loss=0.5294, accuracy=0.7078**

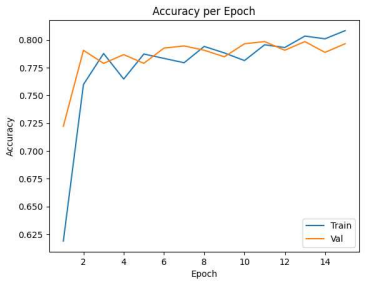
Model Evaluation on Test Data: The Receiver Operating Characteristic (ROC) curve illustrates the model's ability to discriminate between diabetic and non-diabetic cases. The



(a) Config 1: Val_Acc = 0.6908

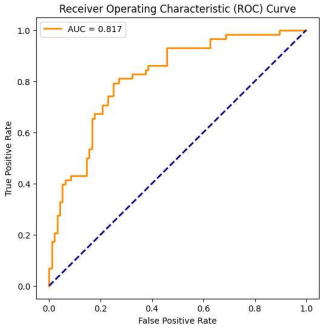


(b) Config 2: Val_Acc = 0.8023

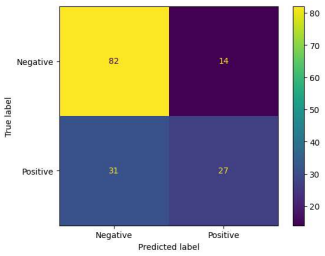


(c) Config 3: Val_Acc = 0.8141

Figure 1: Comparison of validation accuracy across three configurations.



(a) ROC - AUC



(b) Confusion matrix

Figure 2: Test Data Evaluation

Area Under the Curve (AUC) value of 0.817 indicates a good overall classification performance, meaning that the model correctly ranks positive (diabetic) instances higher than negative ones about 81.7% of the time (Fawcett, T 2006, p. 861-874). A perfect classifier would achieve an AUC of 1.0, while 0.5 represents random guessing, indicating that the proposed MLP demonstrates strong discriminative capacity.

The confusion matrix summarizes prediction outcomes on the unseen test set. Out of all non-diabetic samples, 82 were correctly predicted (True Negatives) and 14 were misclassified (False Positives). For diabetic cases, 27 were correctly identified (True Positives) while 31 were missed (False Negatives). This indicates that the model achieves high specificity but comparatively lower sensitivity, a common trade-off in medical prediction tasks where class imbalance exists (Hanley, J.A. & McNeil, B.J. 1982, p. 29-36).

Overall, the MLP with GELU activation achieves reliable generalization with balanced performance across classes, supported by the high AUC and acceptable confusion matrix distribution.

References

[1] Diabetes Australia (n.d.) *Support for people living with diabetes.* Available at: https://www.diabetesaustralia.com.au/, Viewed: 2 November 2025.

[2] Fawcett, T. (2006). An introduction to ROC analysis. *Pattern Recognition Letters*, 27(8), 861–874.

[3] Hanley, J.A. and McNeil, B.J. (1982). The meaning and use of the area under a ROC curve. *Radiology*, 143(1), 29–36.